Many-core Acceleration for Model Predictive Control Systems

Satoshi Kawakami
Graduate School of Information Science and Electrical Engineering Kyushu University
kawakami@soc.ait.kyushu-u.ac.jp

Akihito Iwanaga
Graduate School of Integrated Frontier Sciences Kyushu University
iwanaga@soc.ait.kyushu-u.ac.jp

Koji Inoue
Faculty of Information Science and Electrical Engineering Kyushu University
inoue@ait.kyushu-u.ac.jp

ABSTRACT

This paper proposes a novel many-core execution strategy for real-time model predictive controls. The key idea is to exploit predicted input values, which are produced by the model predictive control itself, to speculatively solve an optimal control problem. It is well known that control applications are not suitable for multi- or many-core processors, because feedback-loop systems inherently stand on sequential operations. Since the proposed scheme does not rely on conventional thread-/data-level parallelism, it can be easily applied to such control systems. An analytical evaluation using a real application demonstrates the potential of performance improvement achieved by the proposed speculative executions.

Categories and Subject Descriptors

J.7 [Computers in Other Systems]: Real Time; F.1.2 [Computation by Abstract Decives]: Modes of Computation—Parallelism and concurrency

General Terms

Experimentation, Performance

Keywords

Many-core Processor, Model Predictive Control, Speculative execution

1. INTRODUCTION

Control systems are ubiquitous in real world. This is because wide range of products and facilities, e.g. electronic devices in house, automobile, aircraft, robots, manufacturing equipment of generators and petrochemical plant, stand on automatic control technologies. For instance, modern automobiles are full of automatic controllers, fuel saving, emission reduction, driver assist such as antilock brake system, etc. It is obvious that improvements of control systems play an important role to make people's lives more prosperous.

Figure 1 shows a simple example of feedback control systems consisting of a controller, a plant (controlled object), a sensor, and an actuator. The input values of the controller denoted as reference $r(t)$ is an objective state of the plant and the sensed data denoted as measured output $x(t)$ indicates the current state of the plant. The output of the controller $u(t)$ is fed to the actuator for controlling the plant. One of the main objective in this control is to minimize the difference between $r(t)$ and $x(t)$ in real time.

There are two major methods for feedback controls, MAP and PID (proportional-integral-derivative). The former decides system input $u(t)$ by referring to a look-up table, and the later computes them based on the difference between the measured output and the reference values. Although these conventional approaches are well used in real control systems, they can not be effectively applied to complex nonlinear systems. MPC (Model Predictive Control) is another promising candidate to achieve efficient control systems, and has come to draw a lot of attention in recent control technology area. The model used in MPC is generally intended to represent the behavior of complex dynamical system (or plant) by means of predicting the change in the measured output $x(t)$. This kind of approach provides an ability to tolerant to system dynamics. Although MPC has high potential to be used in wide range of applications, the most serious issue is the additional complexity of its algorithm. In order to predict the behavior of the plant system, it is required to solve optimization problems for each sensor output. This is the main reason why the major applications of MPC has so far been in use in the process industries such as chemical plants and oil refineries which require second-order real-time operations. Since advanced non-linear complex applications such as automatic driving systems and submarine cable laying ship tend to require millisecond-order effective controls [5], providing ultra high-performance MPC execution platform is a big challenge.

To tackle this issue, this paper proposes a novel execution model of MPC applications on many-core processors. It attempts to exploit effectively the computation resources (i.e. cores) by means of considering the theory of MPC. As far as we know, this is the first paper which tries to accelerate

MES '13, June 23 - 24 2013, Tel-Aviv, Israel

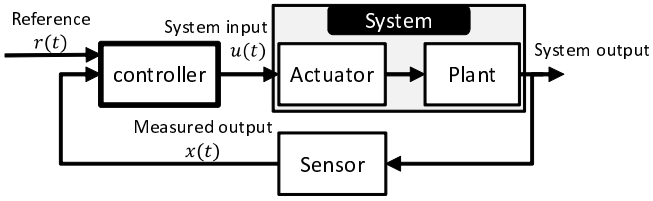

Figure 1: Block diagram of feedback control systems.

the MPC execution by many-core. The contributions of this paper are as follows.

- A bottleneck analysis of MPC executions is performed in order to clarify the limitation of conventional parallel approaches which attempt to exploit data-/thread-level parallelism.
- A novel many-core execution model for MPC is proposed. The key idea behind this approach is to predict measured output in advance and speculatively execute MPC processes. Since MPC is a control methodology which attempts to predict the plant behavior, we can easily obtain one or more candidates for measured output which will be fed to the controller in future. By means of exploiting the abundant cores, the proposed approach starts MPC computations for all of the candidates.
- We analyze the potential of proposed scheme by using a real control application, and show that the upper bound of speedup achieved by the speculation is about 50X. We also report that much higher speedup can be achieved if the target system does not require so high accuracy of computation.

The organization of this paper is as follows. In Section 2, the theory and implementation of MPC are explained so as to clarify the sequential bottleneck. Section 3 presents the proposed speculative execution in detail, and Section 4 discusses the performance potential. Section 5 explains related work, and finally Section 6 concludes this paper.

2. MPC (MODEL PREDICTIVE CONTORL)

2.1 Theory

MPC is a control method which determines system inputs based on the prediction of future plant behavior. Figure 2 explains the theory of MPC. The measured output which represents the current state of the plant is fed to the controller in every sampling time. It is required for the controller to calculate the associated system input until the next measured output arrives, otherwise it fails required real-time operations. The most important feature of MPC is that an optimal control problem is solved by considering the plant's future after a time period called prediction horizon T.

For instance, at time t_1 in Figure 2, the controller gets the measured output $x(t_1)$, and the execution of the MPC process starts to calculate its associated optimal output $u(t_1)$. In this process, the controller attempts to find the optimal

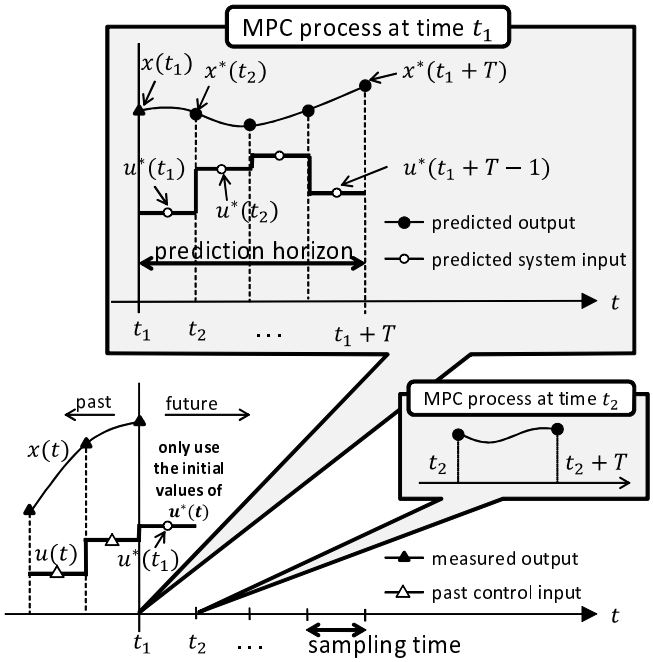

Figure 2: Theory of model predictive control.

set of system inputs denoted as $u^*(t_1), u^*(t_2), ..u^*(t_1+T-1)$, which makes the plant transit to the reference state in minimum time, by predicting the value of measured output denoted as $x^*(t_2), x^*(t_3), ..x^*(t_1+T)$. Finally, $u^*(t_1)$ is used as $u(t_1)$. At the next time t_2, $u(t_2)$ is calculated in a similar way for the prediction horizon $[t_2, t_2+T]$.

2.2 Formulation

MPC can precisely control the behavior of the non-linear systems by accurately formulate a dynamic model of the plant. MPC determine system input by solving the optimal control problem for each sampling period. The authors consider the general nonlinear system to be controlled object as following formula:

$$\dot{x}(t) = f(x(t), u(t), t) \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is state vector $u(t) \in \mathbb{R}^m$ is vector of system input. Using (1), the optimal control problem is presented as in the following formula:

minimize
$\phi(x^*(t+T), t+T) + \int_t^{t+T} L(x^*(\tau), u^*(\tau), \tau) d\tau$
subject to
$\dot{x}^*(\tau) = f(x^*(\tau), u^*(\tau), \tau)$
$x^*(t) = x(t)$
$x_{min} \leq x^*(\tau) \leq x_{max}$
$u_{min} \leq u^*(\tau) \leq u_{max}$

where $\tau (t \leq \tau \leq t+T)$ is variable, $x^*(\tau)$ and $u^*(\tau)$ is variable function. The objective function is constructed by termination cost and interval const. When the state value at a certain time t is given, the objective function is minimized

at $t+T$ and evaluation interval $[t, t+T]$. $x^*(\tau)$ and $u^*(\tau)$ is the state values and the system input in constraints.

2.3 Sequential Bottleneck

First, MPC has a large computational complexity. Optimal control problem of MPC boil down to quadratic programming problems (LQ problem) in general. The computational complexity of the quadratic programming problem with a variable number of N is $O\left(N^3\right)$ [7]. Therefore, with the increasing complexity of the control object, the real-time processing is difficult.

Second, the authors discuss difficulty about parallelizing the numerical calculation in MPC by the traditional data-/thread-level parallelism. The predicted system input $u^*(t)$ is discretised by means of a suitable following partition:

$$t = \tau_0 < \tau_1 < ... < \tau_n = t + T \tag{2}$$

The prediction horizon T is divided into n subintervals $[\tau_i, \tau_{i+1}]$, $0 \leq i \leq n-1$. On each subinterval, we compute the trajectories $x_i(t)$ as the solution of the following ODE (Ordinary Differential Equation) :

$$\dot{x_i}(t) = f(x_i(t), u_i(t), t) \quad \forall t \in [\tau_i, \tau_{i+1}] \tag{3}$$

$$x_i(\tau_i) = s_i \tag{4}$$

where s_i is the initial values in $[\tau_i, \tau_{i+1}]$. Note that every $x_i(\tau_i)$ calculate s_{i+1} using s_i. Therefore, the numerical calculation in MPC is sequential processing and, can not be parallelized. In the case of the pendulum control (section 4.2) , the execution time of this sequential processing account for about 70% of the entire execution time.

3. SPECULATIVE MPC EXECUTION WITH INPUT VALUE PREDICTION

3.1 Concept

In on-chip parallel processing, the degree of parallelism impacts significantly on performance improvement in general. In feedback-loop control systems, however, sequential operations dominate the entire process as explained in Section 2.3. Another well known way to effectively exploit the abundant hardware resources in many-cores is to exploit data-level parallelism, and is suitable for multimedia applications such as movie compression/decompression, image recognition, etc. Since measured output values in feedback-loop systems which are usually outputs of several kinds of sensor devices embedded in the target plant as explained in Figure 1 are fed to a controller in chronological order, this feature exactly prohibits system designers from trying to use such optimization. These two limitations are fundamental reasons why many-core processors can not easily be applied to such control systems.

In order to overcome these limitations and to expand applicability of many-cores, this paper proposes a novel many-core execution strategy to realize real-time optimal control systems. MPC applications have the following two features.

- Particular processes are executed in time series independently. This means that each process can start its execution when the associated input values for the controller, i.e. the measured output values from the plant, are ready.

- In each process, MPC predicts accurately the behavior of target plant, i.e. the measured output values from the plant, then find the optimum strategy to control the actuator.

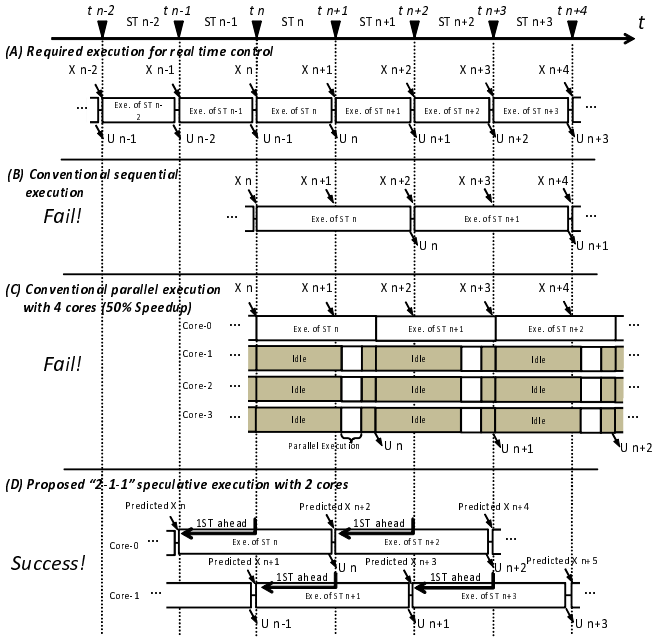

Figure 3: Overview of MPC executions with conventional and proposed approach

These two facts make us to figure out an idea to accelerate MPC executions, temporal parallel processing with speculative executions. As explained above, MPC inherently has an ability to know input values for executions, which will be fed in future. The main idea behind our approach is to exploit the predicted input values not only for control optimization but also for process-level speculative executions.

3.2 Temporal Parallel Processing with Speculative Executions

Execution of each process in real-time MPC has to be completed before the next input arrives as shown in Figure 3(A). Here, X_n shows the input at time t_n for the controller. ST_n and U_n are sampling-time window and the output to the actuator associated with X_n. As explained in Section 2.1, an optimization problem is solved in each sampling-time window. Figure 3(B) shows the case for conventional sequential executions. In this scenario, we assume that the execution of each sampling-time window takes almost double of the time required as the constraint. Although it is possible to apply a conventional thread-level parallel executions, we can not expect dramatic performance improvement due to the lack of parallelism. Although the example of 4-core thread-level parallel executions depicted in Figure 3(C) produces 50%

speed-up, still the performance improvement is not enough to achieve real-time operations.

Figure 3(D) presents a conceptual behavior of the proposed approach assuming dual cores. We represent the features of our speculative executions by a set of three numbers, x-y-z.

- x: The total number of cores to be used for the execution.
- y: The number of candidates (predicted input values) to be used for redundant speculative executions for each process. The detail will be explained in Section 3.4.
- z: The degree of speculation in terms of the number of sampling-time windows. This shows how long in advance each process can start its execution before the associated real input arrives.

Since the speculation degree in Figure 3(D) is one, the execution of sampling-time window n (ST_n) starts at t_{n-1} on core-0. Although the real input value for ST_n is not available at that time, we use a predicted input value for X_n which has been computed in a previous sampling-time window. Core-1 similarly starts the execution for ST_{n+1} at time t_n by using the predicted input value X_{n+1}.

MPC algorithm predicts the plant behavior for several sampling-time windows at each process. This means that the execution of ST_n, ST_{n+1}, and ST_{n+2} predicts X_{n+3} independently for controlling the actuator in each sampling-time window. Since the most recently predicted value tends to be more accurate, we have decided to choose the latest available value for the speculation. For instance, in Figure 3(D), if the speculative execution of ST_{n+2} has updated the predicted value of X_{n+3} until t_{n+2}, it is used for the speculation of ST_{n+3}. Otherwise the predicted value of X_{n+3} from ST_{n+1} is selected.

In our approach, each process is executed on a single core, so that its execution time is the same as the conventional sequential strategy showed in Figure 3(B). Since the execution in Figure 3(D) can be started one sampling-time window before, however, effective execution time becomes to be almost half, resulting in 2X speedup over the conventional single core execution. This performance improvement makes it possible to satisfy the real-time constraints, i.e. outputs U_n are generated as well as the required execution behavior presented in Figure 3(A).

3.3 Exploiting Many Cores for Aggressive Speculations

As explained in Section 3.2, MPC predicts the behavior of plant in several sampling-time windows future. Potentially, the proposed approach can start a speculative execution when the first prediction of associated input value is performed. For instance, in Figure 2, the input value on time $t_1 + T$ (i.e. $x^*(t_1 + T)$) is predicted by solving the optimal control problem in the sampling-time window on t_1, and is updated by following processes until $t_1 + T$. This means that the sampling-time window for $t_1 + T - 1$ can be executed speculatively at any time after t_1. This feature gives

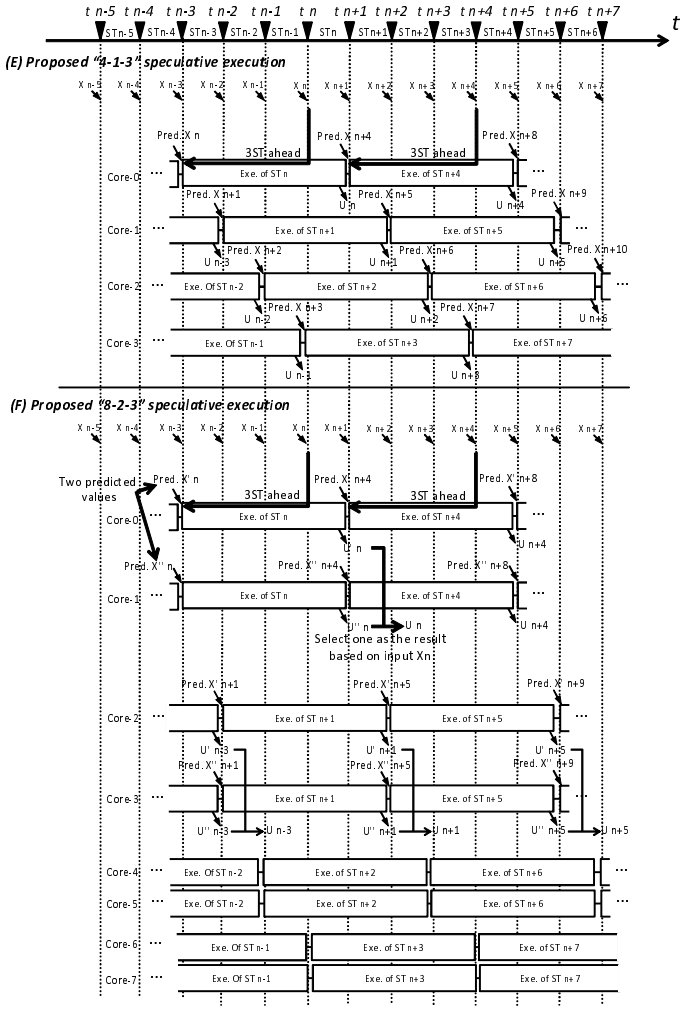

Figure 4: Overview of MPC aggressive, accurate speculations

us a chance to increase the degree of speculation, parameter z. Figure 4(E) shows an execution overview of 4-1-3 proposed speculative execution. In this figure, we assume a more performance-critical MPC application than Figure 3. Since the constraint of sampling time is half, 2X speedup achieved by the proposed execution (D) in Figure 3 fails real-time operations. By using four cores and starting the execution of each process at three sampling-time windows in advance, as showed in Figure 4(E), we can achieve 4X speedup so as to satisfy the real-time constraint. If we assume a perfect prediction, theoretically, we can achieve a perfect scalability which is one of the most important metrics of many-core systems.

3.4 Exploiting Many Cores for Accurate Speculations

Although we can expect high accuracy of input value predictions, increasing the degree of speculations tends to make the predictions inaccurate. To compensate for this negative effect, we propose a redundant speculation which is denoted

by the parameter y explained in Section 3.1. Figure 4(F) depicts the case for 4-2-3 redundant speculations. As we see in the figure, each sample-time window is speculatively executed by two cores, e.g. ST_n is executed on Core-0 and Core-1 in parallel. The difference between the two speculations is only the predicted input data used, X'_n and X''_n. One of the execution results, X'_n or X''_n, is selected based on the associated real input value X_n obtained at time t_n.

Assuming that targets of this paper are digital control systems in which the measured values are discrete, the set of predicted input values for a sample-time window can be expressed as follows:

$$X_{t_{n+1}} = \{x^*{}_{LRC}(t_n) \pm Rd | d, D \in \mathbb{N}^0, 0 \leq d \leq D\} \quad (5)$$

where $x^*{}_{LRC}(t_n)$ is the most recently predicted values, R is rounding unit, and D (dispersion) is the difference between real input value and the predicted value corresponding it. D is indicator of program's character. For example, the case of $D = 0$ shows that real input value and the predicted value is an exact match, the case of $D = 1$ indicates that the difference between the input value and the predicted value is range of $\pm 1 * R$.

In order to discuss the accuracy of input value predictions, when the real input value and the predicted value is an exact match, a hit state is defined as follows:

$$hit(t, R) = \begin{cases} 1 & (Round(x_{\text{diff}}(t), R) = 0) \\ 0 & (otherwise) \end{cases} \quad (6)$$

$$x_{\text{diff}}(t) = \min_{x_{pred}(t) \in X_t} |x_{pred}(t) - x_{obser}(t)| \quad (7)$$

where $x_{\text{diff}}(t)$ is the mininum absolute values of the difference between obserbed values and predictived values at time t. Using (6), hit_rate is defined as follows:

$$hit\_rate(N, R) = \frac{\sum_{t=0}^{N} hit(t, R)}{N} \quad (8)$$

where N is total number of sampling time windows. N is the total simulation time divided by sampling time.

3.5 Handling Miss Speculation

In feedback-loop systems, we can assume that rapid, wide variations in the plant state rarely takes place. This is because the target systems to be controlled has continuous features in many cases. This is one of the reasons why MPC works very well even if for non-liner complex systems. This means that in most cases the proposed speculative executions are accurate.

However, how to handle miss-speculations is an important consideration. The most simple way is to ignore miss predictions and allow inefficient system control, e.g. a long delay until the plant becomes to be stable at the reference state. If the target applications accept this negative effects, this implementation makes the highest performance. The second option is to take a hybrid organization with a conventional MAP approach. If the prediction is correct the execution results obtained from MPC are used, otherwise the MAP table is looked-up. This strategy seems to be reasonable, because we can avoid unpredictable control caused by miss predictions.

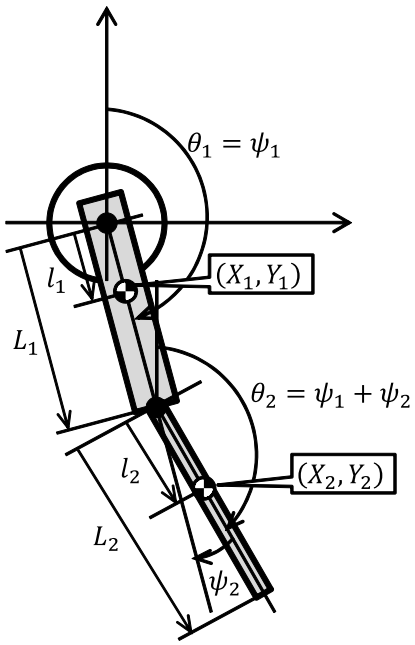

Figure 5: arm typed pendulum swing-up control system (ψ_i: relative angle, θ_i: absolute angle).

4. PRELIMINARY EVALUATION AND ANALYSIS

4.1 Methodology

One of the considerable discussion points for the proposed temporal parallel processing is a tradeoff between the redundancy of executions (parameter y) and the depth of speculation (parameter z). Although we can expect higher performance by increasing the parameter z, it negatively affects to the accuracy of input-value prediction. In order to compensate for this side effect, it may be forced to increase the parameter y. Since occupying large amount of cores for redundant executions limits the potential performance, it is not clear how much we can achieve performance improvement.

To answer this question, an analytical evaluation using a real MPC application, an arm-typed pendulum swing-up control system [6], is performed. We measure the difference of predicted and measured outputs, and then the relation between the number of cores and performance is analyzed. In this evaluation, we assume that the target system requires calculation accuracy of one, tow, or three digit(s) after the decimal point, and the parameter y is decided to satisfy this condition.

4.2 Arm-typed Pendulum Swing-up Control

This section describes the detail of optimal control problem for the arm-typed pendulum swing-up system [6]. Figure 5 shows the pattern diagram of arm-typed pendulum swing-up control system. The initial state of the two arms is the position at The two arms are hanging down as the initial state ($\theta_1 = \theta_2 = \pi$), and the MPC tries to make them stand upright as the target state ($\theta_1 = \theta_2 = 0$) .

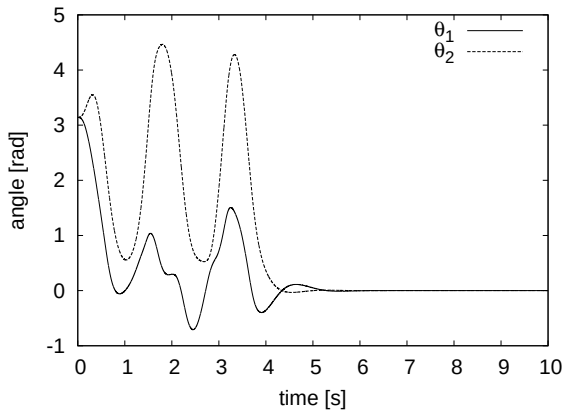

Figure 6: The angle of arms in experimental result

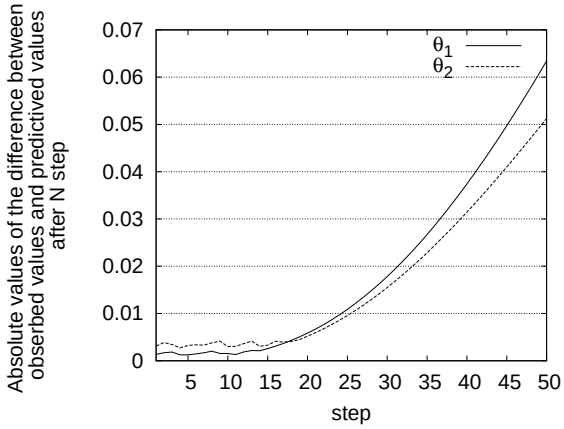

Figure 7: Absolute values of difference between measured and predicted outputs

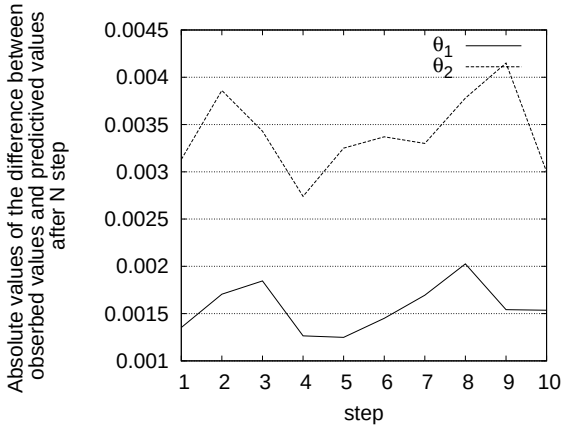

Figure 8: Absolute values of difference between measured and predicted outputs (enlarged)

Here, we formulate the equation of the state. The center of gravity in each arm (X_1, Y_1) is presented as in the following formula:

$$\begin{cases} X_1 = l_1 \sin\theta_1 \\ Y_1 = l_1 \cos\theta_1 \end{cases}, \begin{cases} X_2 = L_1 \sin\theta_1 + l_2 \sin\theta_2 \\ Y_2 = L_1 \cos\theta_1 + l_2 \cos\theta_2 \end{cases} \quad (9)$$

The kinetic energy W, the potential energy U, and the loss energy D are presented as follows:

$$\begin{cases} W = \sum_{i=1}^{2} \left\{ \frac{1}{2} m_i \left(\dot{X}_i^2 + \dot{Y}_i^2 \right) + \frac{1}{2} J_i \dot{\theta}_i^2 \right\} \\ U = \sum_{i=1}^{2} m_i g Y_i, D = \sum_{i=1}^{2} \frac{1}{2} \mu_i \dot{\psi}_i^2 \end{cases} \quad (10)$$

where m_i is the molarity of each arm, J_i is the inertia moment around the center of gravity, g is the acceleration of gravity, and μ_i is viscous friction coefficient. In the optimal control problem based on (9) and (10) , the objective function J is presented as in the following equation:

$$J = \frac{1}{2} x^T(t+T) S_f x(t+T) + \int_t^{t+T} \left(\frac{1}{2} x^T(\tau) Q_x(\tau) + \frac{r_1}{2} u^2(\tau) - r_2 v(\tau) \right) d\tau \quad (11)$$

where the state values x is $\left[\theta_1 \ \theta_2 \ \dot{\theta}_1 \ \dot{\theta}_2\right]^T$, S_f and Q is the positive semi-definite matrices, r_1 and r_2 is positive real number. The prediction horizon T is $0.5[s]$, and the sampling period is $1[ms]$.

The constraint condition is presented as in the following formula:

$$u^2 + v^2 - u^2{}_{max} = 0 \quad (12)$$

where u is input voltage to the motor, and has constraint $|u| \leq u^2{}_{max}$. In this program, u_{max} is $2.5[V]$. v is dummy input for the constraint.

Figure 6 shows the simulation result executed on a 2.93 GHz Intel Xeon 5670. By lowering the pendulum, the angle of arm converges to the target state in about 5 seconds.

4.3 Accuracy of Input-Value Prediction

We measure the difference of absolute values between measured and predicted outputs for the arm-typed pendulum swing-up system. Figure 7 shows experimental results. It is clear from the figure that the difference becomes large with the increase in the depth of speculation (parameter z). The differences are relatively small in the range from 1 to 18, and then suddenly increases. Figure 8 is an enlarged version of Figure 7. We can see from the figure that 9 ($\pm$ 4 plus 1) of redundant execution (parameter y) are needed to guarantee the quality of control inputs which are fed to the actuators if the system require the accuracy of three digits after the decimal point.

4.4 Potential of Performance Improvement

In this paper, we assume that the difference of absolute values between measured and predicted outputs for the arm-

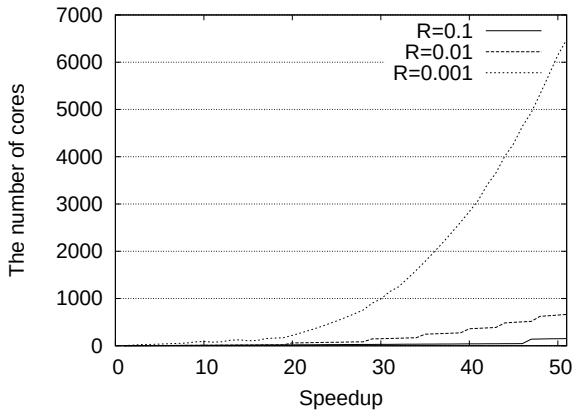

Figure 9: Relation of core-count and speedup.

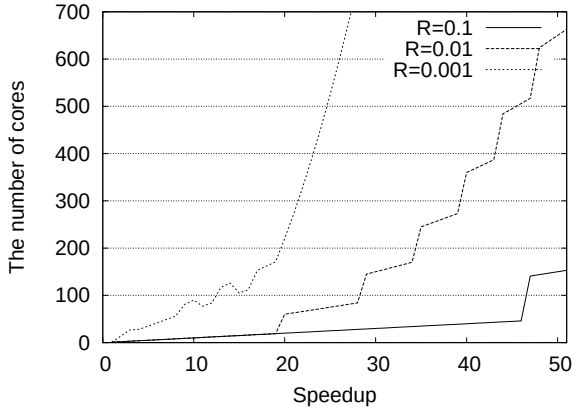

Figure 10: Relation of core-count and speedup (enlarged).

typed pendulum swing-up system is equal to figure 7 in any situation. The number of cores required to ensure the quality of control inputs (i.e. (8) is 100%) can be estimated. Figure 9 shows the core-performance relation for the proposed approach. The x-axis shows speedup requirements in order to satisfy real-time operations, and y-axis gives us the number of cores to be used for achieving the corresponding speedup. R denotes that the required digits after the decimal point. From this figure, we see that the speedup limit is around 50 if the system require the three digits accuracy after the decimal point. On the other hand, the performance scales very well when R is 0.1. Figure 10 is an enlarged version of Figure 9. If we assume a 128 many-core system, the proposed approach has a potential to achieve 46X, 28X, and 16X speedup if R is 0.1, 0.01, and 0.001, respectively.

4.5 A Case Study: 2.93GHz Xeon

Figure 11 shows the execution time of each MPC process on a Xeon processor. The target application requires 1 ms sampling time as explained in Section 4.2. Since the worst case execution time is 7.266$[ms]$, it is necessary to improve the performance of eight times. This means that 8, 8, and 56 cores are needed for $R = 0.1$, 0.01, and 0.001 real-time operations, respectively.

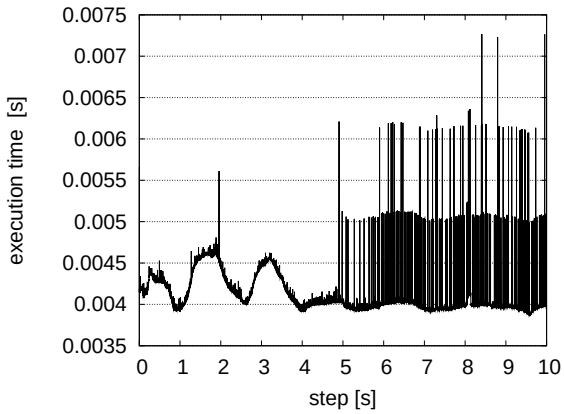

Figure 11: The execution time of each sampling time.

5. RELATED WORK

There are a number of researches to apply MPC to real systems [2] [6]. Since MPC can treat with non-linear complex systems, this approach has a potential to become to be popular in the field of control systems.

Very few number of researches, which aim to accelerate solving the optimal control problem, have so far been reported. Some researchers tried to improve MPC algorithm itself. Although [5] introduces a formulation that can compute directly a stationary solution, its applicability is low, because this approach can adapt only in a special issue such as a linear quadratic control problem. [7] [4]attempt to reduce computational complexity by means of considering efficient numerical calculations. [3] discusses how to accelerate MPC executions by focusing on parallel matrix calculations. As explained in Section2.3, unfortunately, feedback-loop systems inherently consists of sequential operations. Therefore, we can not expect drastic performance improvements due to this limitation.

A remarkable advantage of embedded many-core processors is that aggressive on-chip parallel executions can achieve both high performance and low power consumption at the same time, because the clock frequency and supply voltage can be reduced without sacrificing the overall performance by means of exploiting parallelism existing in target applications. This means that the most important challenges are (1) ensuring a sufficient level of parallelism and (2) maintaining performance scalability against the increased number of cores. One of the killer applications for many-core processors is media processing such as movie compression, decompression, and recognition, because they inherently have large amount of parallelism, e.g. from pixel (or data) level to frame (or thread) level. Therefore, a number of researchers focused on such applications [1][8]. Unlike these conventional researches, out target application is control systems in which extracting thread-/data-level parallelism is very hard.

6. CONCLUSIONS

In this paper, we have proposed a novel many-core execution model to accelerate MPC. The key idea is to exploit predicted measured output, which are produced by the model

predictive control itself, and speculatively execute an optimal control problem. The result of the preliminary experiment using a real MPC application clearly shows the potential of outstanding performance improvement.

Our ongoing work is to increase the number of benchmark programs. Then, we implement the proposed scheme in a real many-core processor for detailed performance evaluations. Since the power consumption is the first order of design constraints in recent embedded systems, we consider a low power technique for real-time MPC applications in the future.

7. ACKNOWLEDGMENTS

We thank Professor Toshiyuki Ohtsuka of Kyoto University for providing MPC applications. He also gave us worthwhile advices through private technical meetings.We would like to thank all members of the System LSI Lab. of Kyushu University.

8. REFERENCES

[1] S. Bell and B. e. a. Edwards. Tile64 - processor: A 64-core soc with mesh interconnect. In Proc. IEEE International Solid-State Circuits Conference, pages 88 –598, Feb. 2008.

[2] H. Ferreau, G. Lorini, and M. Diehl. Fast nonlinear model predictive control of gasoline engines. In Proc. IEEE International Conference on Control Applications, pages 2754 –2759, Oct. 2006.

[3] R. Gu, S. Bhattacharyya, and W. Levine. Methods for efficient implementation of model predictive control on multiprocessor systems. In IEEE International Conference on Control Applications (CCA), pages 1357 –1362. IEEE, Sept. 2010.

[4] T. Ohtsuka. Continuation/gmres method for fast algorithm of nonlinear receding horizon control. In Proc. IEEE Conference on Decision and Control, volume 1, pages 766 –771. IEEE, 2000.

[5] O. Toshiyuki. Introduction nonlinear optimal control in japanese. corona, 2011.

[6] O. Toshiyuki. Model predictive contorl (development, feature : control engineering in the illustration for beginners) in japanese. system/control/information : The Institute of Systems Control and Information Engineers Journals, 56(6):310–312, Jun 2012.

[7] Y. Wang and S. Boyd. Fast model predictive control using online optimization. IEEE Trans. Control Systems Technology, 18(2):267 –278, March 2010.

[8] H. Xu, J. Tanabe, H. Usui, S. Hosoda, T. Sano, K. Yamamoto, T. Kodaka, N. Nonogaki, N. Ozaki, and T. Miyamori. A low power many-core soc with two 32-core clusters connected by tree based noc for multimedia applications. In Symposium on VLSI Circuits (VLSIC), pages 150 –151, June 2012.